CLAIMS

1. A computer-implemented method for a Φ function providing a mechanism for single static assignment in the presence of predicated code, the method comprising the steps of:

introducing an associated ordered guard on each source operand in a control or predicate Φ instruction;

materializing a Φ function by inserting at least one copy from each source operand to a target variable in the same order as said source operand; and

predicating each of said copies by said ordered guard associated with said source operand.

2. The method of Claim 1, further comprising:

transforming a source code by writing a result of a compare operation on a variable in said source code to a predicate;

representing said transformed source code in static single assignment form using said Φ function having source operands;

materializing said Φ function; and

eliminating any unnecessary copies from said source operands.

3. The method of Claim 1, further comprising the steps of:

ordering said source operands according to a topological ordering of the source code blocks; and

maintaining said topological ordering through any subsequent code transformations.

4. The method of Claim 3, wherein said topology is determined by a compiler.

5. The method of Claim 4, further comprising the steps of:

the compiler taking a stream of said source code;

the compiler identifying the blocks and edges of said source code; and

the compiler topologically numbering said blocks.

6. The method of Claim 1, comprising the step of:

inserting a predicate Φ function after each existing predicated assignment.

7. The method of Claim 6, wherein said predicate Φ function is constructed during the initial construction of single static assignment form.

8. The method of Claim 6, wherein said guard on said predicate Φ functions indicates a predicate under which said associated source operand is live.

9. The method of Claim 6, wherein said predicate Φ function is constructed while already in static single assignment form.

10. The method of Claim 1, further comprising the step of either replacing or augmenting a control Φ function with a predicate Φ function.

11. The method of Claim 10, wherein said guard on said control Φ functions indicates the basic block which is the source of the edge associated with said source operand.

12. The method of Claim 1, wherein said ordered guards indicate the condition under which an associated source operand is live.

13. A computer-implemented method for a Φ function providing a mechanism for single static assignment in the presence of predicated code, the method comprising the steps of:

transforming a source code by writing a result of a compare operation on a variable in said source code to a predicate;

representing said transformed source code in static single assignment form using a Φ function having source operands;

introducing an associated ordered guard on each source operand in a block of said source code;

ordering said source operands according to a topological ordering of the source code blocks;

maintaining said topological ordering through any subsequent code transformations;

materializing said Φ function by inserting at least one copy from each source operand to a target variable in the same order as said source operand; and

eliminating any unnecessary copies from said source operands.

14. A system for a Φ function providing a computer-implemented mechanism for single static assignment in the presence of predicated code, comprising:

a transforming module accessible by said computer for transforming a source code by writing a result of a compare operation on a variable in said source code to a predicate;

a single static assignment module accessible by said computer for representing said transformed source code in static single assignment form using a Φ function having source operands;

an ordered guard module accessible by said computer for introducing an associated ordered guard on each source operand in a block of said source code;

a compiler for topologically ordering said blocks of said source code;

an ordering module accessible by said computer for maintaining said topological ordering through any subsequent code transformations;

a materializing module accessible by said computer for materializing said Φ function by inserting at least one copy from each source operand to a target variable in the same order as said source operand; and

an eliminating module accessible by said computer for eliminating any unnecessary copies from said source operands.